UDACITY

PROJECT

Predicting Boston Housing Prices

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

4 SPECIFICATIONS REQUIRE CHANGES

This is a very impressive submission. Just need a few minor adjustments and you will be golden, but also check out some other ideas presented in this review. One tip here would be that some of these topics are extremely important as you embark on your journey throughout your Machine Learning career and it will be well worth your time to get a great grasp on these topics before you dive deeper in. Keep up the hard work!!

Data Exploration

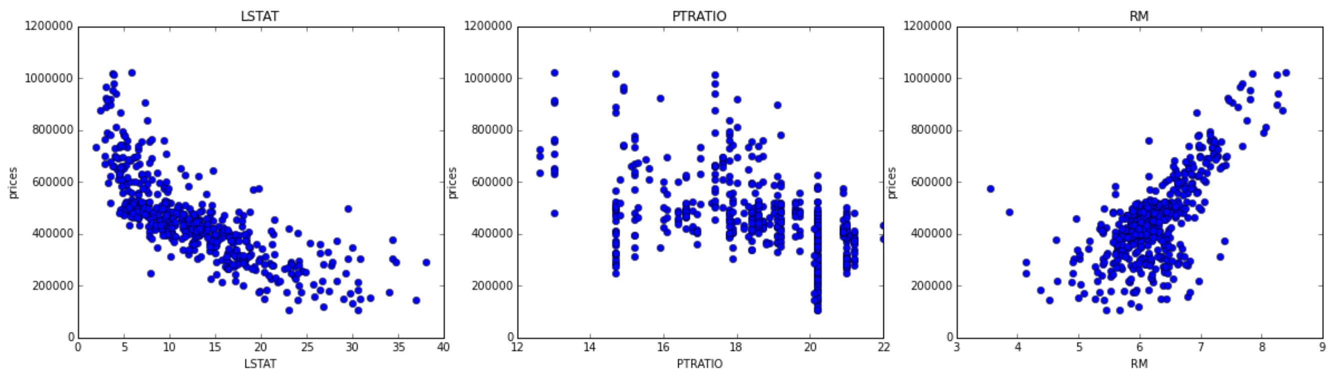
All requested statistics for the Boston Housing dataset are accurately calculated. Student correctly leverages NumPy functionality to obtain these results.

Good job utilizing the power of Numpy!! Always important to get a basic understanding of our dataset before diving in. As we now know that a "dumb" classifier that only predicts the mean would predict $454,342.94 for all houses.

Student correctly justifies how each feature correlates with an increase or decrease in the target variable.

Great idea to check out the correlation coefficients! As we can confirm these ideas by plotting each feature vs MEDV housing prices.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 5))
for i, col in enumerate(features.columns):
    plt.subplot(1, 3, i)
    plt.plot(data[col], prices, 'o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('prices')
```



Developing a Model

Student correctly identifies whether the hypothetical model successfully captures the variation of the target variable based on the model’s R^2 score.
The performance metric is correctly implemented in code.

Good ideas! In general, the higher the R-squared, the better the model fits your data. So with a high value of 92.3% (0.923) we can clearly see that we have strong correlation between the true values and predictions.

Another great idea with regression problems is to examine a residual plot. A residual plot is a graph that shows the residuals on the vertical axis and the independent variable on the horizontal axis. If the points in a residual plot are randomly dispersed around the horizontal axis, a linear regression model is appropriate for the data; otherwise, a non-linear model is more appropriate.

Student provides a valid reason for why a dataset is split into training and testing subsets for a model. Training and testing split is correctly implemented in code.

Awesome reasoning here. Since we need a way to determine how well our model is doing! This is very intuitive! As we can get a good estimate of our generalization accuracy on this testing dataset. Since our main goal is to accurately predict on new unseen data. Can also note that we can try and protect against overfitting with this independent dataset.

If you would like to learn some more ideas in why we need to split our data and what to avoid, such as data leakage, check out these lectures

- https://classroom.udacity.com/courses/ud730/lessons/6370362152/concepts/63798118300923
- https://classroom.udacity.com/courses/ud730/lessons/6370362152/concepts/63798118310923

Analyzing Model Performance

Student correctly identifies the trend of both the training and testing curves from the graph as more training points are added. Discussion is made as to whether additional training points would benefit the model.

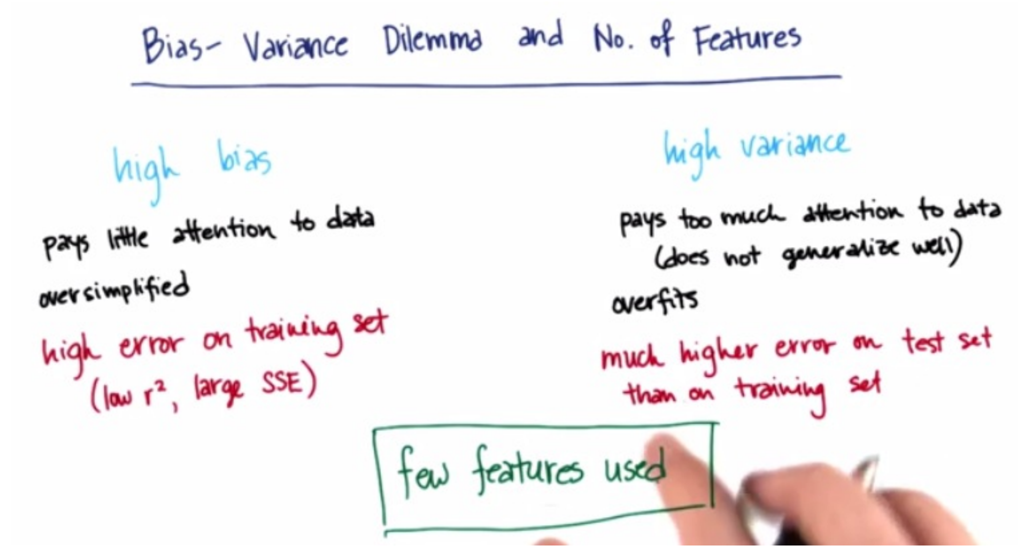
You are correct with your comment of " *as more and more data added after reaching 250, the slope of two curves doesn't change much.*" As if we look at the testing curve here, we can clearly see that it has actually converged to its optimal score, so more data is not necessary.

However also for this section make sure you describe the initial trends of the training and testing curves before they converge, for your chosen max depth of 3. What do the initial phases look like(increasing or decreasing)?

Student correctly identifies whether the model at a max depth of 1 and a max depth of 10 suffer from either high bias or high variance, with justification using the complexity curves graph.

Solid visual justification! Could also simply mention the large gap for high variance. You clearly understand the bias/variance tradeoff.

- As a max_depth of 1 suffers from high bias, visually this is due to the low training and validation scores(also note that it has low variance since the scores are close together). As this model is not complex enough to learn the structure of the data
- And a max_depth of 10 suffers from high variance, since we see a large gap between the training and validation scores, as we are basically just memorizing our training data and will not generalize well to new unseen data


Bias- Variance Dilemma and No. of Features
high bias
pays little attention to data
oversimplified
high error on training set
(low r², large SSE)
high variance
pays too much attention to data
(does not generalize well)
overfits
much higher error on test set
than on training set
few features used

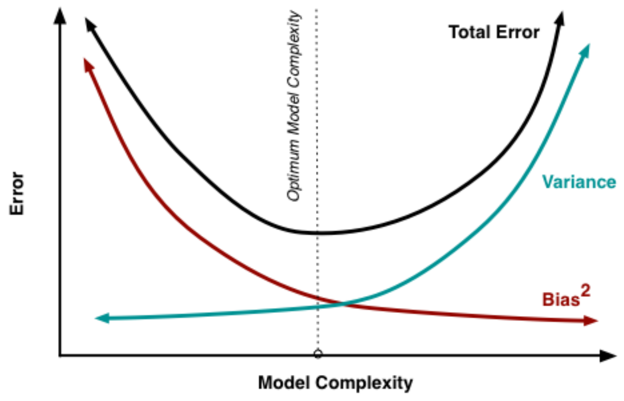
Student picks a best-guess optimal model with reasonable justification using the model complexity graph.

"*As the complexity curve shows, the gap between traing score and validation score is the least when max_depth=3, meaning that it may be the best point for bias-variance tradeoff.*"

This is a good choice here. Just note that you should also mention that the scores are high as well to depict the *bias* part of the bias - variance tradeoff. But nice job!

As we are looking for the highest validation score(which is what gridSearch searches for). And we are also looking for a good bias / variance tradeoff(with close training and validation scores).

Check out this visual, it refers to error, but same can be applied to accuracy(just flipped)



Evaluating Model Performance

Student correctly describes the grid search technique and how it can be applied to a learning algorithm.

You are correct that gridSearch is a parameter optimization technique. Therefore lastly for this section please also mention which hyper-parameter value combinations does it test (i.e. a random sample of them, every other combination, all of the exhaustively)? What are these "*different combinations of hyperparameters*"?

Links

- (http://scikit-learn.org/stable/modules/grid_search.html)
- (https://en.wikipedia.org/wiki/Hyperparameter_optimization#Grid_search)

Student correctly describes the k-fold cross-validation technique and discusses the benefits of its application when used with grid search when optimizing a model.

You are off to a good start here, but need a bit more analysis

- in terms of the k-fold cross-validation technique make sure you also mention the last step. How we get that one last final score from k-fold cross-validation? What do we do with all the error rates at the end of the K-Training?
- can you also expand a bit more in terms of the benefits when used with grid search when optimizing a model. Why does this "*improves the efficiency*"? Why is it beneficial to utilize all the data for training and testing purposes? Why can't we do parameter tuning on the testing set from a simple train/test split? This is a crucial topic and will be well worth the time and effort to get a solid grasp on it now.

Links

- (https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation)
- Video
- (https://www.cs.cmu.edu/~schneide/tut5/node42.html)

Student correctly implements the `fit_model` function in code.

You have correctly implemented GridSearchCV. Would also recommend setting a `random_state` in your DecisionTreeRegressor for reproducible results.

```
regressor = tree.DecisionTreeRegressor(random_state = "any number")
```

Student reports the optimal model and compares this model to the one they chose earlier.

Congrats!

Student reports the predicted selling price for the three clients listed in the provided table. Discussion is made for each of the three predictions as to whether these prices are reasonable given the data and the earlier calculated descriptive statistics.

You on off to a good start here, but please expand on your justification for these features. Why are these **features** reasonable? As a good idea would be to go client by client and give an idea for each feature of `RM`, `LSTAT`, and `PTRATIO`.

Optional: A more advanced and great idea would be to compare these to the descriptive stats of the features. We can compute the five number summary of the descriptive stats of the features with

```
features.describe()
```

Student thoroughly discusses whether the model should or should not be used in a real-world setting.

Very nice ideas. Would agree, as this dataset is quite old and probably doesn't capture enough about housing features to be considered robust!

Love the extra coding and great idea to check out the testing accuracy, as maybe a model such as a random forrest could improve these results?

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Student FAQ